M68000 -MOTOROLA'S SWEET SIXTEEN

News has been coming in thick and fast recently about Motorola's forthcoming addition to the new range of 16-bit super micros, the M68000. But with something approaching a six month wait before any sort of general availability, has it arrived too late? Nicholas Jarmany largely dodges that question and instead casts an appreciative eye over its capabilities.

The Intel 8086 was the first of the new 16-bit micros to appear, closely followed by the Zilog Z8000. At the moment there is still no physical sign of Motorola's contender, so presumably Intel and Zilog are rubbing their hands with glee. The only dampener for them is that the M68000 is almost certainly the most powerful of the three; in fact at one stage when some of the big manufacturers saw the advance specifications, it was said that Motorola just wouldn't be able to make it. It now looks, however, as if the scepticism was ill-founded for sample devices are already spreading round the world.

Internal Operation/Layout

The M68000 internal structure is that of a 32-bit micro, making it very efficient with long word operations. There are 17 32-bit registers (apart from a 32-bit program counter and a 16-bit status register) comprising eight data registers for 8, 16 and 32 bit data and seven address registers. All 17 registers can be used as index registers and there is also a specific user and supervisor stack pointer.

There are two modes of operation, user and supervisor. In user mode certain instructions are illegal and areas of memory can be locked out by a memory management unit. When in this mode a switch to supervisor mode always occurs when an interrupt, bus error etc is received. In supervisor mode all instructions are available and the full status register can be accessed. This arrangement is similar to that of the Z8000.

A trace mode can be set in supervisor mode which causes a branch via a trace vector after execution of every instruction — very useful for program debugging! The lower 512 words of memory are reserved for a vector table containing 255 vectors, of which 192 are reserved for user interrupt vectors.

Interrupts, bus errors etc. all cause what Motorola calls 'exception processing' , of which there are three levels of priority. In order of decreasing priority, Group 0 contains — Reset (highest), Bus Error, Address Error; Group 1 — Trace, Interrupt, Illegal Instruction and Privilege Violation; Group 2 (all equal priority) — TRAP, TRAPV, CHK, Zero Divide. All the exceptions cause branching via the appropriate vector, except for certain occurrences of Bus Error. If a Bus Error and a Halt signal are received simultaneously, the processor will re-run the current memory access on the negation of $\overline{\text{HALT}}$.

Instruction set

There are 56 basic instruction types and 14 addressing modes, and although this doesn't seem like many instructions, it's deceptive as there are many variations. For example MOVE caters for loading register(s), storing register(s), moving data in memory etc. The total number of useful instructions exceeds 1000! The addressing modes are extremely comprehensive and no programmer could envisage needing more. The format of the instructions is astonishingly simple and easy to use. With other micros you have to learn the code for each individual instruction — e.g. Load register (indexed) might be 0A and load register (immediate) FE. Not so with the M68000 . . . All you need to learn are the numbers for the 56 basic instructions and the numbers for the addressing modes. The complete instruction is then made up of the code for the instruction, the data size, addressing mode and register number (if required). Dead simple!

Speed

The speed of the M68000 is also something to be marvelled at. It's faster than the 8086, the Z8000 and the PDP11/45 — and it can't be a lot slower than the PDP 11/70! It's twice as fast as the Z8000 on a 16-bit multiply (35 instruction cycles compared with 70 — maximum).

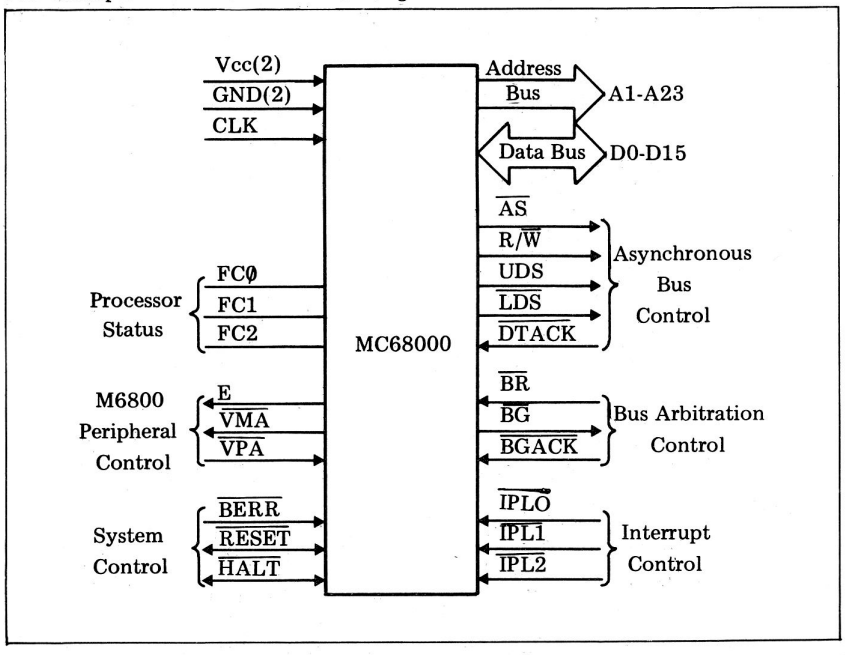
Omissions

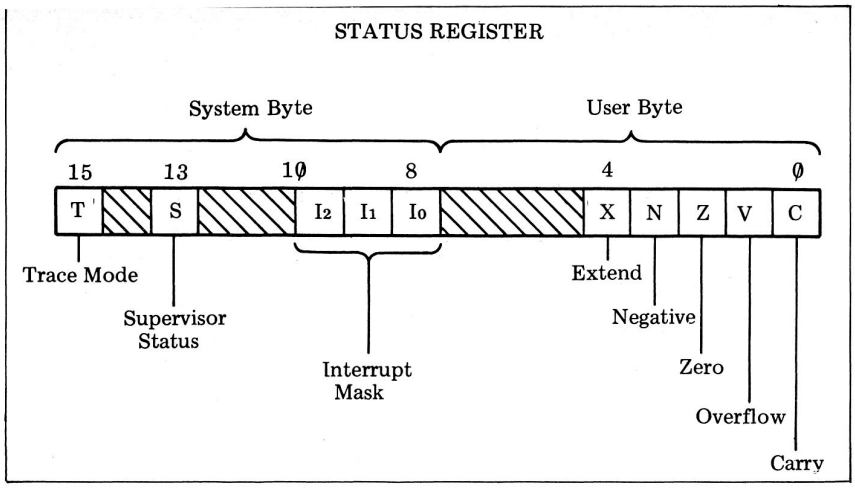
Unlike the Z8000 the M68000 does not have on-chip refresh and multi-micro control. It could be that Motorola does not want to be seen to be abandoning its traditional approach in favour of

FUNCTION CODE OUTPUTS

FC2	FC1	FC0	Cycle Type
L	L	L	(Undefined Reserved)
L	L	H	User Data
L	H	L	User Program
L	H	H	(Undefined Reserved)
H	L	L	(Undefined Reserved)
H	L	H	Supervisor Data
H	H	L	Supervisor Program
H	H	H	Interrupt Acknowledge

L = Low H = High





somebody else's. I feel, however, that there has been a preference towards getting as much computing power into the CPU as possible — at the expense of other features that can easily be added on with a few external chips. (Try extending an instruction set with a few external chips!) What Motorola has aimed at is producing the most powerful single chip CPU in the world. Possibly the plan has succeeded.

Hardware

The M68000 has definitely been designed for large systems, although a small system could easily be based around it. The processor is contained in a 64-pin package (long!) which is needed because none of the signals are multiplexed, thus increasing speed and ease of use. It requires +5V and a single phase clock (up to 8MHz). . . an internal cycle is defined as two clock cycles (250ns). There are 23 address lines giving 16 Mbytes of direct addressing; individual bytes are accessed via the $\overline{UDS}$ and $\overline{LDS}$ signals. A valid address is indicated by $\overline{AS}$ and the addressed device responds with $\overline{DTACK}$ (Data Transfer ACKnowledge). This also acts to stretch memory cycles (if necessary) by not being negated until the memory is ready. Memory read, write, read-modify-write cycles take 4, 5 and 9 clock cycles respectively. A great feature of the M68000 is its ability to interface directly with standard M6800 peripherals. If, at the beginning of a memory access cycle, a $\overline{VPA}$ signal is received, the processor switches to the M6800 form of addressing. $\overline{VMA}$ is taken low and E is equivalent to M6800 Φ2 (1MHz). Hence $\overline{VPA}$ can be derived from the address decoding logic on any M6800 peripheral boards.

FC0 to FC2 are outputs that show the type of processing currently being done in the CPU; $\overline{IPL0}$ to $\overline{IPL2}$ are inputs devoted to interrupts. Seven levels of interrupt are available (level 0 = no interrupt), level 7 being the highest priority. With all seven levels of interrupt the vector address for the service routine can either be supplied by the interrupting device — or else an autovector can be used. To my knowledge this is the most advanced form of interrupt handling available on a micro. On reception of an interrupt, an interrupt acknowledge code is placed on FC0-FC2 and a read cycle is entered with the interrupt level on the lower three bits of the address bus. The processor then expects the vector address to be placed on the data bus and $\overline{DTACK}$ to be given. If this does happen then the processor jumps to the location pointed to by the contents of

DATA ADDRESSING MODES

Mode	Generation
Register Direct Addressing	
Data Register Direct	EA = Dn
Address Register Direct	EA = An
Absolute Data Addressing	
Absolute Short	EA = (Next Word)
Absolute Long	EA = (Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	EA = (PC) + d_{16}
Relative with Index and Offset	EA = (PC) + (Xn) + d_8
Register Indirect Addressing	
Register Indirect	EA = (An)
Postincrement Register Indirect	EA = (An) An ←An + N
Predecrement Register Indirect	An ←An - N, EA = (An)
Register Indirect With Offset	EA = (An) + d_{16}
Indexed Register Indirect With Offset	EA = (An) + (Xn) + d_8
Immediate Data Addressing	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	
Implied Register	EA = SR, USP, SP, PC

NOTES:
EA Effective Address
An Address Register
Dn Data Register
Xn Address or Data Register used as Index Register
SR Status Register
PC Program Counter
( ) Contents of
d_8 Eight-bit Offset (displacement)
d_{16} Sixteen-bit Offset (displacement)
N 1 for Byte, 2 for Words and 4 for Long Words
← Replaces

INSTRUCTION SET

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
BCC	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DBCC	Test Cond, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
SCC	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

the memory location whose address is on the bus. If this does not occur then the CPU assumes an autovector and jumps using the autovector corresponding to the interrupt level.

BERROR is an input that can signify a non-responding device or an illegal access determined by an external memory management chip. The effect this signal has depends on certain conditions already described. Both the RESET and the HALT lines are bi-directional, allowing external devices to be reset via the reset instruction. An internally generated halt is caused when a Bus Error signal is received on two consecutive memory accesses. When this occurs an externally generated reset is required to restart the CPU. This feature provides useful protection in the event of a catastrophic system failure!

Summary

The strong points of the M68000 seem to be its simple, easy to learn instruction format and its useful range of instructions (including control of both stacks and queues), coupled with the ability to maintain linked stacks. With floating point instructions on the way, writing high level languages will be a piece of cake! This sort of instruction should enable more efficient programming and the introduction of many mainframe techniques.

Bus arbitration is also very comprehensive, allowing simple control of a multi-master bus. The direct interface to M6800 peripherals must appeal to a lot of people, as it will mean that most of their existing equipment could easily be used in a M68000 system, thus eliminating a lot of annoying and expensive duplication of costs.

And yet all this extra power results

GOTO page 119

VARIATIONS OF INSTRUCTION TYPES

Instruction Type	Variation	Description
ADD	ADD	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	AND	Logical And
	ANDI	And Immediate
CMP	CMP	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	EOR	Exclusive Or
	EORI	Exclusive Or Immediate
MOVE	MOVE	Move
	MOVEA	Move address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	NEG	Negate
	NEGX	Negate with Extend
OR	OR	Logical Or
	ORI	Or Immediate
SUB	SUB	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

Motorola's Sweet Sixteen Continued from Page 99

in no extra difficulties in system design. More and more of the complications of circuit design seem to be disappearing into fewer LSI and VLSI chips.

Motorola has made it clear that it expects to extend the instruction set in the near future, to include instructions like FIX and FLOAT (floating point to integer and vice versa); it also expects to bring out a 16MHz version. But what about now? Small quantities of the M68000 are expected on the market in the next few months, but at the moment all the chips are going to the big firms for evaluation — not surprisingly in view of the new competition in this extremely valuable market. However, as soon as second-sources get into production the supply position is bound to improve. I was told by one of Motorola's distributors that they had achieved 98% functional chips from the very first masks — an astounding achievement for a chip of this complexity!

A memory management chip is also mentioned in the advance spec on the M68000 but no one, not even in the labs at Motorola UK, could tell me anything about it, so I doubt if it can be appearing at all this year. But who cares? If you can get your hands on an M68000 you'll be too heavily occupied to think about anything else for some time to come!

My thanks to Hawke-Cramer for helping to obtain information for this article. Technical details are based on data derived from Motorola Advanced Specification Data Sheets.